## AMENDMENTS TO THE SPECIFICATION

Please amend the specification of the present application as set forth below. Changes to the specification are shown by strikethrough (for deleted matter) and underlining (for added matter).

Please replace the paragraph beginning on page 3, line 6, with the following rewritten paragraph:

-- Large-scale web applications, which are accessible worldwide, are particularly susceptible to development delays caused by the need to make the applications functional for many different cultures, languages, and regions of the world. Currently, localizing applications for many different locales involves brute force replicating of multiple versions of the same application. These separate versions are either developed independently or the entire application is translated by a translation service. The translation process requires in depth knowledge of both natural languages and web presentation technologies, such as HTML and JAVA SERVER PAGE (JSP) technology. As a result it is neither cost efficient nor very reliable. This replicating approach is also expensive when updating the application, as multiple versions of each change are required. --

Please replace the paragraph beginning on page 10, line 15, with the following rewritten paragraph:

-- One example of an interaction-based model is a command bean model that employs multiple discrete program modules, called "Command Beans", that are called for and executed. The command bean model is based on the ~~"Java Bean"~~ JAVA BEAN technology from Sun Microsystems, which utilizes discrete ~~Java~~™ JAVA technology

program modules. Other examples of an execution model include an action-view model and a use case model. The action-view model employs action handlers that execute code and provide a rendering to be served back to the client. The use case model maps requests to predefined UML (Unified Modeling Language) cases for processing. --

Please replace the paragraph beginning on page 11, line 16, with the following rewritten paragraph:

-- The data abstraction layer 208 maps the domain object model to the various external resources 108. The data abstraction layer 208 contains the domain framework 250 for mapping the business logic to a specific problem domain, thereby partitioning the business applications and application managers from the underlying domain. In this manner, the domain framework 250 imposes no application-specific semantics, since it is abstracted from the application model. The domain framework 250 also does not dictate any functionality of services, as it can load any type of functionality (e.g., ~~Java™~~ JAVA technology classes, databases, etc.) and be used to interface with third-party resources. --

Please replace the paragraph beginning on page 13, line 4, with the following rewritten paragraph:

-- In this implementation, the presentation layer 212 is divided into two tiers: a presentation tier and a content rendering tier. The request dispatcher 224 implements the presentation tier. It selects an appropriate data type, encoding format, and protocol in which to output the content so that it can be carried over a network and rendered on the client. The request dispatcher 224 is composed of an engine 262, which resides at the framework 220 in the illustrated implementation, and multiple request dispatcher types (RDTs) 264 that accommodate many different data types, encoding formats, and

protocols of the clients. Based on the client device, the engine 262 makes various decisions relating to presentation of content on the device. For example, the engine might select an appropriate data encoding format (e.g. HTML, XML, EDI, WML, etc.) for a particular client and an appropriate communication protocol (e.g. HTTP, ~~Java™~~ JAVA technology, RMI, CORBA, TCP/IP, etc.) to communicate the response to the client. The engine 262 might further decide how to construct the reply for visual appearance, such as selecting a particular layout, branding, skin, color scheme, or other customization based on the properties of the application or user preference. Based on these decisions, the engine 262 chooses one or more dispatcher types 264 to structure the reply. --

Please replace the paragraph beginning on page 18, line 16, with the following rewritten paragraph:

-- Fig. 4 illustrates the compilation and translation system 400 that adapts servable content (e.g., documents, forms, web pages, UI screens, etc.) from one locale to one or more other locales. The compilation and translation system 400 employs a tool, referred to as the "internationalization compiler" 402, which reads documents (e.g., web pages, email forms, UI screens, etc.) of an application authored for one locale and automatically converts those documents into a form that can be easily localized to any other locale through the translation process. The compiler 402 implements a set of rules that dictate how an input language (such as HTML, XML, WML, JSP technology, etc.) shall be localized. For example, an HTML rule defines which attribute value is localized (translated), and how the tag body, if defined, is localized (direct translation, treated as a script, etc.). The compiler 402 is equipped with a parser (or more generally, a content analyzer) to parse the language(s) for the application. As an example, for a J2EE based

application, the parser is able to parse HTML content (including ~~JavaScript~~ JAVASCRIPT technolgy) and JSP technology pages. --